Lab 3 Report –Thumb ISA Gem5

Kunjian Song

Exercise 1

The execution time and dynamic power are shown in the table below against various optimization levels. The results below are executed using only "SimpleMemory".

ARM32		
Optimization_Level	ARM32_Execution_Time_SimpleMem	ARM32_Dymanic_Power_SimpleMem
0	1.227919	0.00188716
1	0.241052	0.00217868
3	0.145352	0.00233308
Thumb		
Optimization_Level	Thumb_Execution_Time_SimpleMem	Thumb_Dynamic_Power_SimpleMem
0	1.315874	0.00186867
1	0.287514	0.00208689
3	0.276062	0.00199102

Table 1: Benchmark execution time and power consumption for ARM32 and Thumb

The optimization levels (-O) can influence the execution time on both ARM32 and Thumb ISA. As for ARM32, dynamic power also increases with -O levels. However, optimization level does not have much impact on power consumption using Thumb ISA.

This trend does not match the simulation statistics generated by Gem5. The code size decreases with increasing optimization level. Also, the total number of instructions and ALU usages decreases when using -O3. But the dynamic power for ARM32 ISA still increases. Based on what we have learnt in the lecture, this might have caused by more memory operations.

Simulation was also run using the "--caches" options for gem5 along with the "SimpleMemory" [1]. The "--caches" option introduces another level of cache, L1d cache, in the microarchitecture being simulated. As we can see in Table 2, the execution time with L1d caches is much less than the simulation without caches.

The "--caches" option will also generate additional data in the statistics report on hit/miss rate, e.g. the total miss L1d miss rate is reported in "system.cpu.dcache.overall_misses::total". It was found that the overall miss rate in L1d cache increases when using a deeper optimization level. This might be the cause to the growing dynamic power.

The trends are plotted in Figure 1 and Figure 2.

ARM32			
Optimization_Level	ARM32_Execution_Time_Cache	ARM32_Dymanic_Power_Cache	ARM32_overall_miss_rate_cache
0	0.090987	0.00534248	0.000481
1	0.018057	0.00919358	0.063452
3	0.010971	0.0111817	0.063429
Thumb			
Optimization_Level	Thumb_Execution_Time_Cache	Thumb_Dynamic_Power_Cache	Tumb_overall_miss_rate_cache
0	0.097473	0.00509403	0.00048
1	0.0215	0.00797952	0.063407
3	0.020653	0.00669502	0.063395

Table 2: Benchmark execution time and power consumption for ARM32 and Thumb

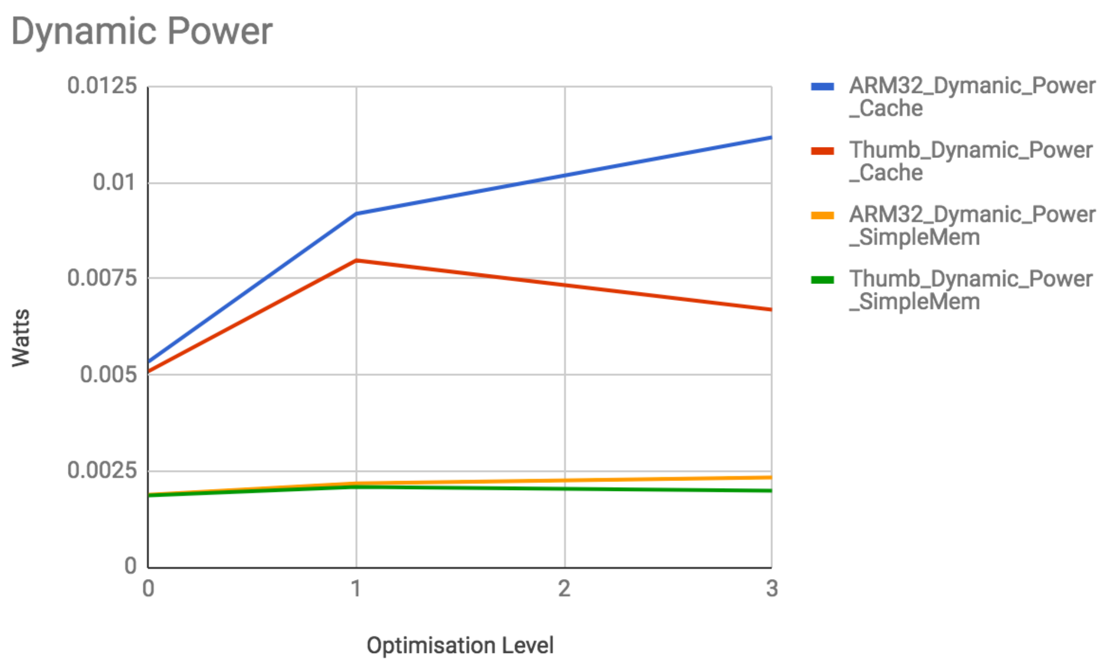

Figure 1: Dynamic Power trends for ARM32 and Thumb with or without L1d cache

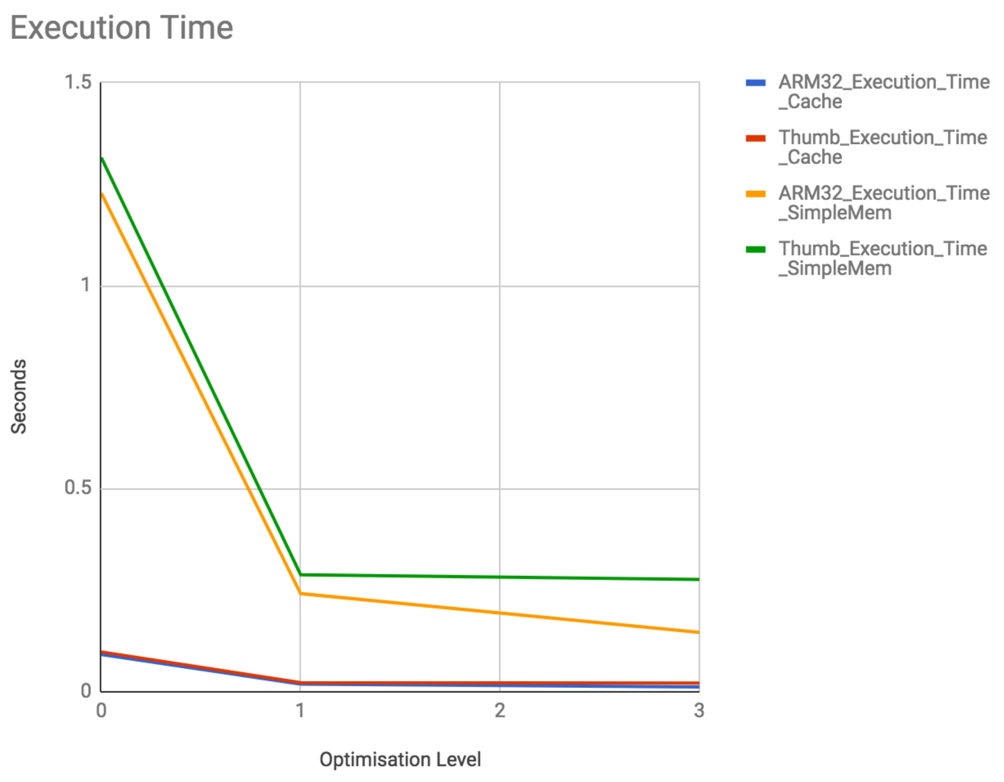

Figure 2: Execution Time for ARM32 and Thumb with or without L1d cache

Exercise 2

The performance/power against clock frequency are shown in Figure 3 and Figure 4.

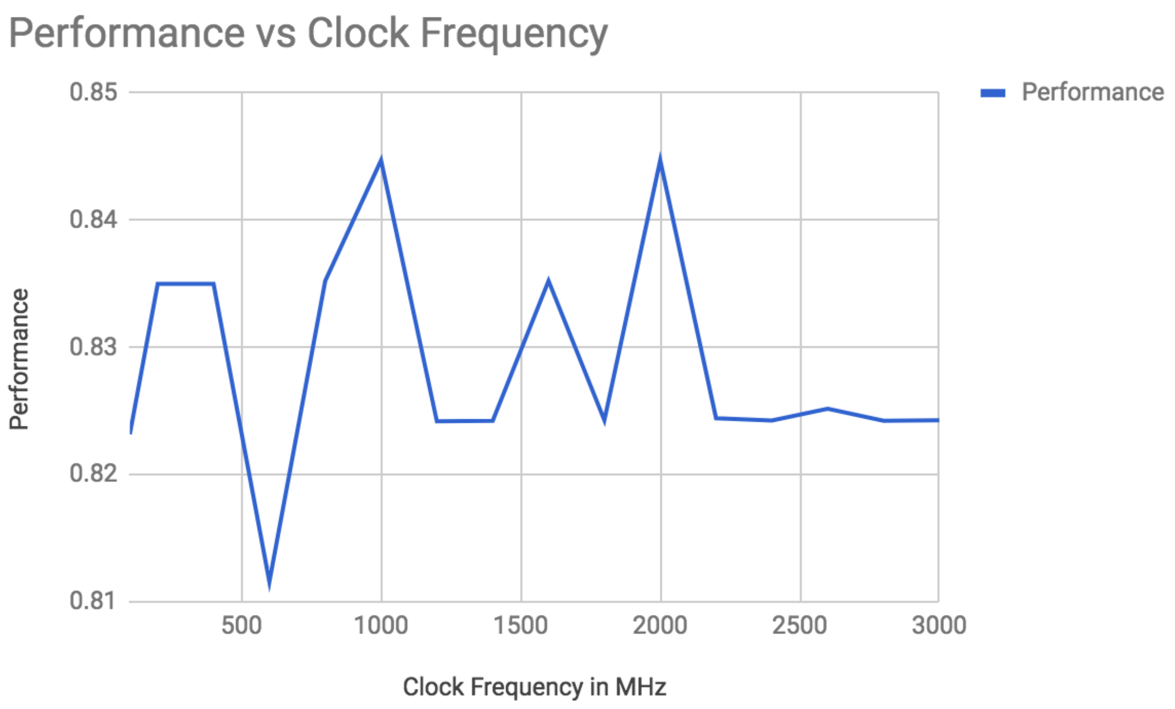

Figure 3: Performance vs. clock frequency plot

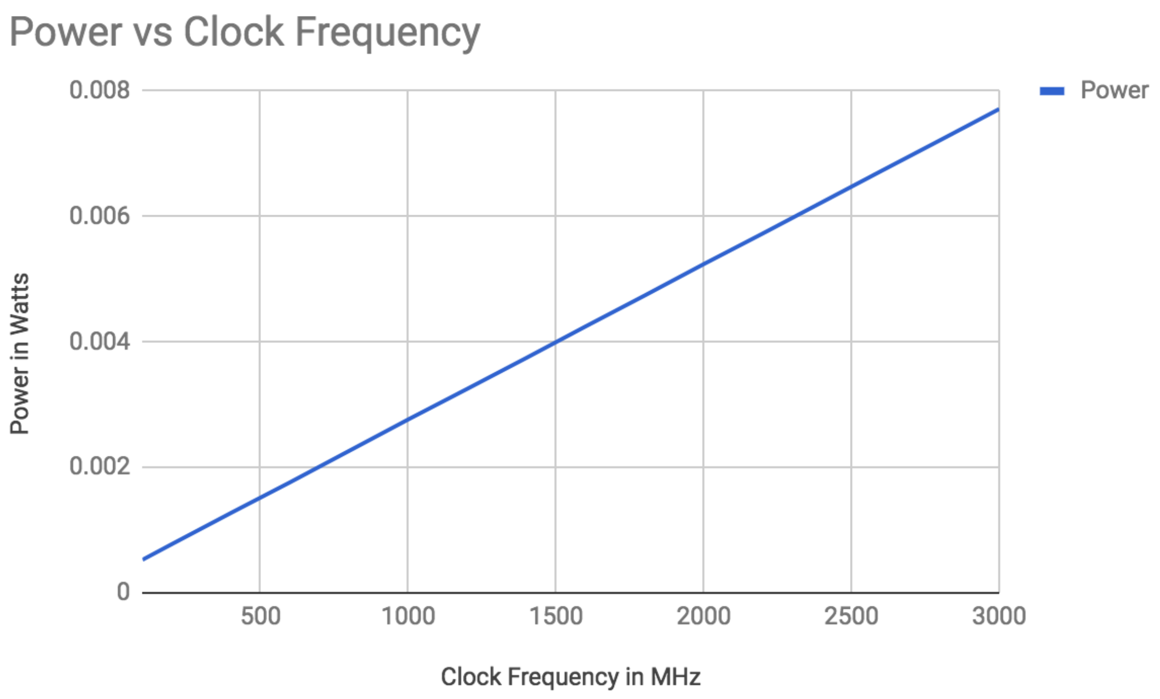

Figure 4: Power vs. clock frequency plot

The power rises linearly with increasing clock frequencies. However, the performance plot has a different trend. Starting with the following classic CPU performance equation from H&P book:

$$Execution\ Time = Number\ of\ Instions\ \times\ CPI\ \times\ Cycle\ Time$$

Where cycle time is basically 1/frequency, and performance is the reciprocal of execution time obtained from the above equation.

Based on the above equation, in order to obtain an accurate execution time, we need to use a cycle-accurate model. However, gem5 is not cycle-accurate [2].

The gem5 simulator is an event-drive simulator [3]. It can model and simulate accurate events and behaviours but not cycle-accurate, meaning that the CPI here may not be accurate, which could in turn give fluctuating lines for performance plot.

Bibliography

[1] “Memory System in gem5,” [Online]. Available: https://www.gem5.org/documentation/general_docs/memory_system/

[2] “Architectural Exploration with gem5,” ARM, [Online]. Available: https://www.gem5.org/assets/files/ASPLOS2017_gem5_tutorial.pdf

[3] “Event-driven programming,” [Online]. Available: http://learning.gem5.org/book/part2/events.html